**PYTHON FOR VISION TECHNIQUES**

**WEEK - 2**

**Name:** Vatsal Panchal

**Registration number:** 18BEC1125

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import cv2

import matplotlib.pyplot as plt

a=cv2.imread('C:/Users/WELCOME/Downloads/download.jpg')

plt.figure()

cv2.imshow('lena',a)

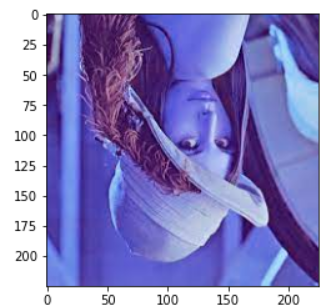
cv2.waitKey(0)

cv2.destroyAllWindows()

flip\_a=cv2.flip(a,0)

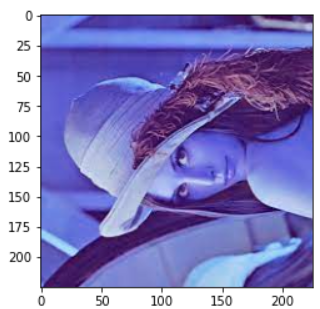
plt.figure()

plt.imshow(flip\_a,cmap='brg')



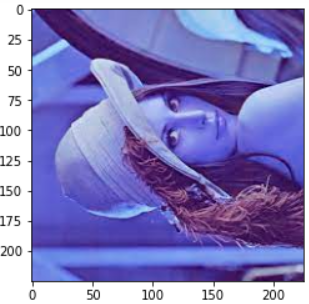
transpose\_a=cv2.transpose(a)

plt.imshow(transpose\_a)



rot\_a=cv2.rotate(a, cv2.ROTATE\_90\_COUNTERCLOCKWISE)

plt.imshow(rot\_a)



print('############# Log Transformation ################')

print('s=c\*log(1+r)')

print('c=255 /log(1 + rmax(a))')

############# Log Transformation ################

s=c\*log(1+r)

c=255 /log(1 + rmax(a))

img=cv2.imread('C:/Users/WELCOME/Downloads/download2.jpg')

a=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

#a=rgb2gray(img)

c=255 / np.log(1 + np.max(a))

#c=1.5

log\_a = c \* (np.log(a + 1))

log\_a=np.array(log\_a, dtype=np.uint8)

#plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.figure(1)

plt.imshow(a, cmap='gray')

plt.figure(2)

plt.hist(a)

plt.title("Histogram of original image")

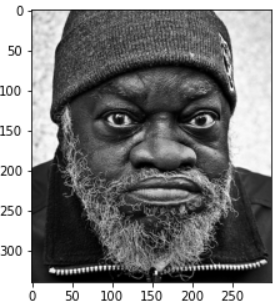
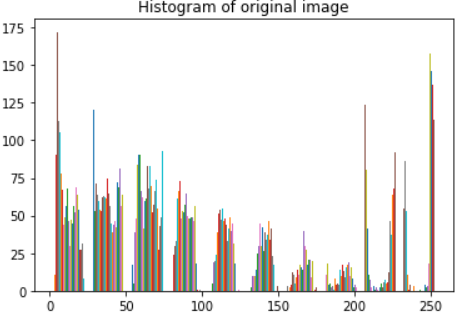
plt.figure(3)

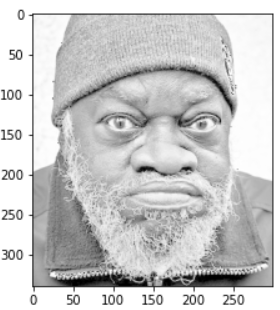
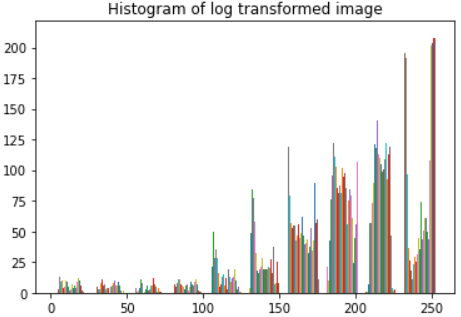
plt.imshow(log\_a, cmap='gray')

plt.figure(4)

plt.hist(log\_a)

plt.title("Histogram of log transformed image")

print('############# brightness ################')

img=cv2.imread('C:/Users/WELCOME/Downloads/download2.jpg')

b=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

#b\_bright=255\*(b/255)+100

#b\_bright=b+100

c=np.full(np.shape(b),100)

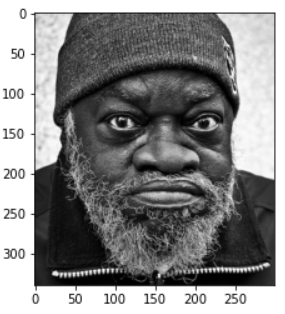
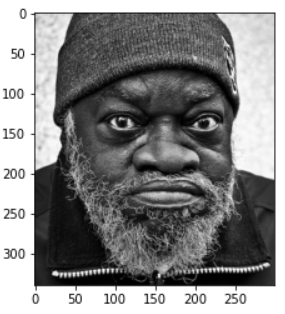
b\_bright=255\*(b/255)+c

plt.figure(1)

plt.imshow(b, cmap='gray')

plt.figure(2)

plt.imshow(b\_bright,cmap='gray')

print('############ Gamma Transformation ###############')

img = cv2.imread('C:/Users/WELCOME/Downloads/download2.jpg')

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

# Trying 4 gamma values.

a=rgb2gray(img)

plt.figure()

plt.imshow(a, cmap='gray')

plt.title("Original image")

for gamma in [0.1, 0.5, 1.2, 2.2]:

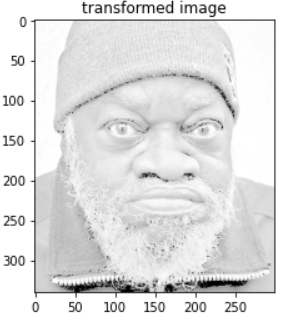
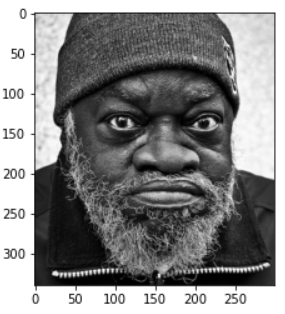
gamma\_corrected = np.array(255\*(a / 255) \*\* gamma, dtype = 'uint8')

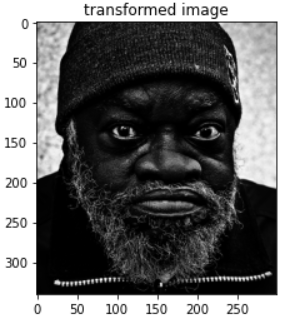
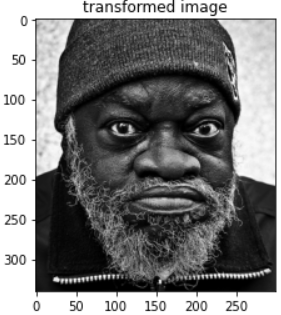
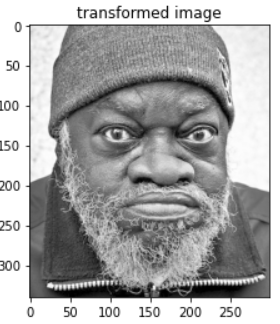
print('gamma\_transformed '+str(gamma)+'.jpg')

plt.figure()

plt.imshow(gamma\_corrected, cmap='gray')

plt.title("transformed image")





print('%%%%%%%%%%%%%%%%% CONTRAST STRETCHING %%%%%%%%%%%%%%%%%%%%%')

img = cv2.imread('C:/Users/WELCOME/Downloads/download3.jpg')

a=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

[row, col]=np.shape(a)

r\_max=np.max(a)

r\_min=np.min(a)

s\_max=255

s\_min=50

b=np.zeros(np.shape(a))

for i in range(row):

for j in range(col):

# if (0 <= a[i][j] and a[i][j] <= r\_min):

# b[i][j]=(s\_min / r\_min)\*a[i][j]

# elif (r\_min < a[i][j] and a[i][j] <= r\_max):

b[i][j]=((s\_max - s\_min)/(r\_max - r\_min)) \* (a[i][j] - r\_min) + s\_min

# else:

# b[i][j]=((255 - s\_max)/(255 - r\_max)) \* (a[i][j] - r\_max) + s\_max

plt.figure(1)

plt.imshow(a, cmap='gray')

plt.figure(2)

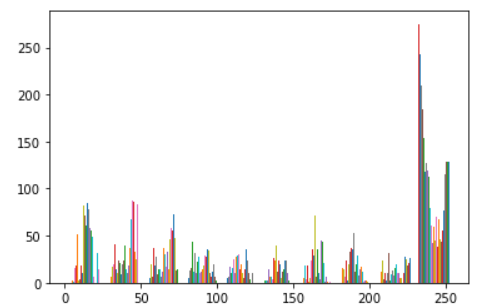
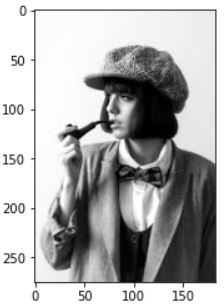
plt.hist(a)

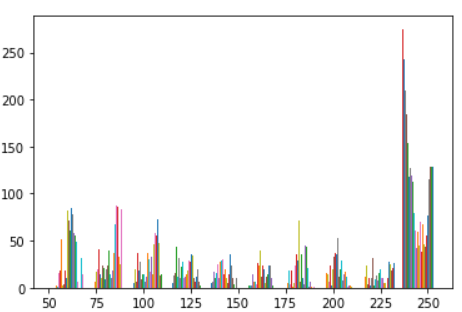
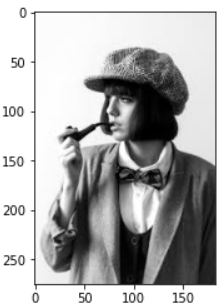
plt.figure(3)

plt.imshow(b, cmap='gray')

plt.figure(4)

plt.hist(b)





**TASK 5- FILTER WORK – SATHIYA SIR**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from scipy import ndimage # Multi-dimensional image processing

img=mpimg.imread('C:/Users/WELCOME/Downloads/IMG\_20200628\_173313.jpg')

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

img=rgb2gray(img)

img\_noise = img + 0.005\* np.random.randn(\*img.shape)

blurred\_img1=ndimage.gaussian\_filter(img\_noise, sigma=1.5) # Multidimensional Gaussian filter

blurred\_img2=ndimage.median\_filter(img\_noise,5) # Calculate a multidimensional median filter

plt.figure(1)

plt.imshow(img,cmap='gray')

plt.title("Original image")

plt.figure(2)

plt.imshow(img\_noise,cmap='gray')

plt.title("Noisy image")

plt.figure(3)

plt.imshow(blurred\_img1,cmap='gray')

plt.title("Gaussian filtered image")

plt.figure(4)

plt.imshow(blurred\_img2,cmap='gray')

plt.title("Median filtered image")

err1=img - blurred\_img1

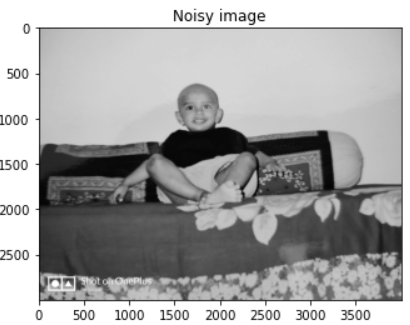
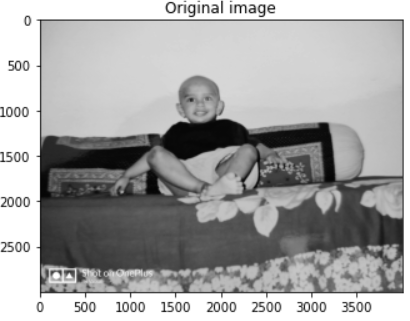
err2=img - blurred\_img2

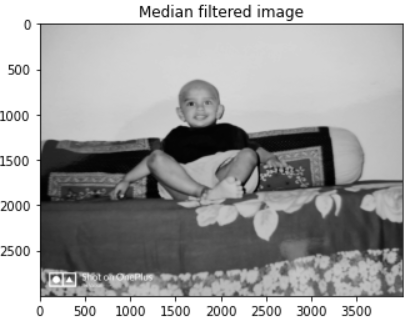
sqerr1 = np.power(err1,2)

sqerr2 = np.power(err2,2)

print(sqerr1.mean())

print(sqerr2.mean())





**MINIMUM AND MAXIMUM FILTERING**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from scipy import ndimage # Multi-dimensional image processing

img=mpimg.imread('C:/Users/WELCOME/Downloads/vatsal.jpg')

def rgb2gray(rgb):

r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

return gray

img=rgb2gray(img)

img\_noise = img + 0.05\* np.random.randn(\*img.shape) # for removing noise change to 0

blurred\_img1=ndimage.maximum\_filter(img\_noise,5) # Multidimensional Gaussian filter

blurred\_img2=ndimage.minimum\_filter(img\_noise,5) # Calculate a multidimensional median filter

plt.figure(1)

plt.imshow(img,cmap='gray')

plt.title("Original image")

plt.figure(2)

plt.imshow(blurred\_img1,cmap='gray')

plt.title("Maximum filtered image")

plt.figure(3)

plt.imshow(blurred\_img2,cmap='gray')

plt.title("Minimum filtered image")

